C 12

```
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
   SSSSSSSSSSSS  MMM         MMM     GGGGGGGGGGG   RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMMMMM   MMMMMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
SSS              MMM   MMM   MMM  GGG              RRR         RRR        TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
   SSSSSSSSS     MMM         MMM  GGG              RRRRRRRRRRRR           TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGGG RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGGG RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG   GGGGGGGGGG RRR   RRR              TTT        LLL
            SSS  MMM         MMM  GGG          GGG RRR      RRR           TTT        LLL
            SSS  MMM         MMM  GGG          GGG RRR      RRR           TTT        LLL
            SSS  MMM         MMM  GGG          GGG RRR      RRR           TTT        LLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
SSSSSSSSSSSS     MMM         MMM     GGGGGGGGG     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$2

Val
---
001
001
001
001
001
001
001
001
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
**FILE**ID**SMGPUTTEX                                   J 13




  SSSSSSSS  MM      MM    GGGGGGGG  PPPPPPPP    UU      UU  TTTTTTTTTT  TTTTTTTTTT  EEEEEEEEEE  XX      XX
  SSSSSSSS  MM      MM    GGGGGGGG  PPPPPPPP    UU      UU  TTTTTTTTTT  TTTTTTTTTT  EEEEEEEEEE  XX      XX
SS          MMMM  MMMM  GG          PP      PP  UU      UU      TT          TT      EE          XX      XX
SS          MMMM  MMMM  GG          PP      PP  UU      UU      TT          TT      EE          XX      XX
SS          MM  MM  MM  GG          PP      PP  UU      UU      TT          TT      EE            XX  XX
SS          MM  MM  MM  GG          PP      PP  UU      UU      TT          TT      EE            XX  XX
  SSSSSS    MM      MM  GG          PPPPPPPP    UU      UU      TT          TT      EEEEEEEE        XX
  SSSSSS    MM      MM  GG          PPPPPPPP    UU      UU      TT          TT      EEEEEEEE        XX
        SS  MM      MM  GG  GGGGGG  PP          UU      UU      TT          TT      EE            XX  XX
        SS  MM      MM  GG  GGGGGG  PP          UU      UU      TT          TT      EE            XX  XX
        SS  MM      MM  GG      GG  PP          UU      UU      TT          TT      EE          XX      XX    ....
        SS  MM      MM  GG      GG  PP          UU      UU      TT          TT      EE          XX      XX    ....
SSSSSSSS    MM      MM    GGGGGG    PP          UUUUUUUUUU      TT          TT      EEEEEEEEEE  XX      XX    ....
SSSSSSSS    MM      MM    GGGGGG    PP          UUUUUUUUUU      TT          TT      EEEEEEEEEE  XX      XX    ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0 MODULE SMG$$PUT_TEXT_TO_BUFFER ( %TITLE 'Put text to display buffer'
0                   IDENT = '1-012'         ! File: SMGPUTTEX.B32 Edit: PLL1012
0                   ) =
1 BEGIN
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1 !
1
1 !++
1 ! FACILITY:     Screen Management
1 !
1 ! ABSTRACT:
1 !
1 !       This is an internal routine used by screen management procedures to
1 !       place user's text into a display buffer.  The text is spanned for
1 !       special characters.
1 !
1 ! ENVIRONMENT:  User mode - AST reentrant
1 !
1 ! AUTHOR: P. Levesque, CREATION DATE: 14-Apr-1983
1 !
1 ! MODIFIED BY:
1 !
1 ! 1-001 - Original.  PLL 14-Apr-1983
1 ! 1-002 - Finish coding.  PLL 20-Apr-1983
1 ! 1-003 - Add error message, character set buffer allocation.  PLL 4-May-1983
1 ! 1-004 - Fix second half of the scan table to agree with actions for
1 !         DEC Multinational.  PLL 5-May-1983
1 ! 1-005 - If on the last line and we have found a line feed, scroll.  PLL 11-May-1983
1 ! 1-006 - If a bell character is found, call SMG$RING_BELL instead of setting
1 !         a bell bit.  PLL 20-May-1983
1 ! 1-007 - If a LF is found, scroll according to the new dcb top & bottom of
1 !         scrolling region fields.  PLL 26-May-1983
1 ! 1-008 - If an ESC is detected, call the terminal simulator routine to
1 !         interpret the sequence and perform the correct SMG$ function.
1 !         PLL 7-Jul-1983
```

```
1 ! 1-009 - Allow 2 'reserved' positions in upper half of table to pass thru
1 !         as printable characters.  PLL 17-Aug-1983
1 ! 1-010 - SMG$$SIM_TERM may set the graphics bit in the DCB's default
1 !         attributes byte.  Take this into account when copying the attribute
1 !         bytes for characters into the buffer.  PLL 29-Aug-1983
1 ! 1-011 - Call SMG$$SIM_TERM when DCB_V_ALLOW_ESC is set.  PLL 2-Sept-1983
1 ! 1-012 - In order to print carriage control characters instead of execute
1 !         them, check the DCB_V_DISPLAY_CONTROLS bit and move the ascii rep
1 !         into the text buffer in a different way.  PLL 23-Sep-1983
1 !--
1
```

```
;     70          0069  1 %SBTTL 'Declarations'
;     71          0070  1 !
;     72          0071  1 ! SWITCHES:
;     73          0072  1 !
;     74          0073  1
;     75          0074  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;     76          0075  1
;     77          0076  1 !
;     78          0077  1 ! LINKAGES:
;     79          0078  1 !
;     80          0079  1 !       NONE
;     81          0080  1 !
;     82          0081  1 ! TABLE OF CONTENTS:
;     83          0082  1 !
;     84          0083  1
;     85          0084  1 FORWARD ROUTINE
;     86          0085  1     SMG$$PUT_TEXT_TO_BUFFER;
;     87          0086  1
;     88          0087  1 !
;     89          0088  1 ! INCLUDE FILES:
;     90          0089  1 !
;     91          0090  1
;     92          0091  1 REQUIRE 'RTLIN:SMGPROLOG';              ! defines Psects, macros, data base
;     93          0169  1
;     94          0170  1 !
;     95          0171  1 ! MACROS:
;     96          0172  1 !
;     97          0173  1 !       NONE
;     98          0174  1 !
;     99          0175  1 ! EQUATED SYMBOLS:
;    100          0176  1 !
;    101          0177  1 !       NONE
;    102          0178  1 !
;    103          0179  1 ! FIELDS:
;    104          0180  1 !
;    105          0181  1 !       NONE
;    106          0182  1 !
;    107          0183  1 ! PSECTS:
;    108          0184  1 !
;    109          0185  1
;    110          0186  1 !
;    111          0187  1 ! EXTERNAL REFERENCES:
;    112          0188  1 !
;    113          0189  1
;    114          0190  1 EXTERNAL ROUTINE
;    115          0191  1     SMG$$SIM_TERM,
;    116          0192  1     SMG$$SCROLL_AREA,
;    117          0193  1     SMG$RING_BELL;
;    118          0194  1
;    119          0195  1 EXTERNAL LITERAL
;    120          0196  1     SMG$_FATERRLIB,
;    121          0197  1     SMG$_STRTERESC;
;    122          0198  1
;    123          0199  1 ! Some constants needed by reference.
;    124          0200  1 OWN
;    125          0201  1     ALLONES     : BYTE INITIAL (-1);
;    126          0202  1
```

```
  1 ! The following macro is used to move a control character into the
  1 ! text buffer in such a way that output will later convert to the
  1 ! appropriate device dependent graphic character.
  1
  1 MACRO
M 1         $INSERT_CTRL_CHAR (CHAR) =
M 1         BEGIN
M 1         LOCAL
M 1             INDEX,
M 1             REMAINING_COLS;
M 1
M 1         REMAINING_COLS = .DCB [DCB_W_NO_COLS] - .DCB [DCB_W_CURSOR_ROW];
M 1         INDEX = $SMG$LINEAR (.DCB [DCB_W_CURSOR_ROW], .DCB [DCB_W_CURSOR_COL]);
M 1
M 1         IF 1 GTR .REMAINING_COLS
M 1         THEN
M 1             WORK_OVERFLOW = .BYTES_REMAINING
M 1         ELSE
M 1             BEGIN                   ! move the low nibble into the high nibble
M 1             LOCAL
M 1                 SHIFT_NIBBLE : BYTE,
M 1                 WORK_ATTR;
M 1             SHIFT_NIBBLE = (CHAR <0,4>) ^ 4;
M 1             CH$MOVE (1, SHIFT_NIBBLE, TEXT_BUF [.INDEX]);
M 1             WORK_ATTR = ATTR_M_USER_GRAPHIC OR .ATTR_CODE;
M 1             CH$MOVE (1, WORK_ATTR, ATTR_BUF [.INDEX]);
M 1             END;
M 1
M 1         DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] + 1;
M 1         IF .DCB [DCB_W_CURSOR_COL] EQL .DCB [DCB_W_NO_COLS]
M 1         THEN
M 1             DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
  1         END%;
  1
  1 !<BLF/PAGE>
```

```
!+
! The table below (CHAR_TABLE) is used with a SCANC instruction to
! detect characters that have an impact on how text needs to be
! positioned in a text buffer that models what is on a portion of the
! screen.  Each character position is occupied by a code indicating
! the kind of action that this character has on text placement.
! Characters are grouped into 10 categories based on their impact on
! the terminal and hence on their impact on what should be placed in
! the buffer at what position.
!
! These categories (codes) are:
!
!       Action Code      Action
!       -----------      ------
!            0           Normal processing.  Character occupies next
!                        available slot in buffer.  Cursor column is
!                        advanced by 1 after placement.
!
!            1           Character can be discarded.  Cursor is not
!                        advanced.
!
!            2           Character can be discarded.  Cursor is not
!                        modified, but a note must be made that the
!                        bell needs to be sounded.
!
!            3           Character can be discarded, but cursor must be
!                        backed up one column.  Be careful about cursor
!                        already being in column 1.
!
!            4           Character can be discarded, but cursor must be
!                        advanced to next TAB stop and intervening
!                        character positions in the buffer are
!                        undisturbed.
!
!                        TAB stops are assumed to be set in the following
!                        columns with column numbering starting at 1:
!                        9, 17, 25, 33, 41, 49, 57, 65, 73 ( width=80)
!
!                        9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97,
!                        105, 113, 121, 129 ( width=132)
!
!            5           Character can be discarded.  Cursor must be
!                        advanced by one line.
!
!            6           Character can be discarded.  Cursor must be
!                        advanced by one line.  (VT treated the same
!                        as #5, FF.)
!
!            7           Character can be discarded.  Effect is
!                        to clear the buffer and reset the cursor to
!                        line 1 column 1.
!
!            8           Character can be discarded.  Effect is to set
!                        cursor to column 1 of current line.
!
!            9           Character can be discarded.  For this version,
!                        ESC terminates the string.  Eventually, subsequent
```

```
1 !                                   characters need to be inspected to see if they
1 !                                   constitute a recognized escape sequence whose
1 !                                   effect must be simulated-- E.g., cursor setting,
1 !                                   rendition setting.
1 !
1 !                                   Some problems with this are:
1 !                                       1. What to do about sequences that we don't
1 !                                          recognize ?
1 !                                       2. What to do about sequences that we
1 !                                          recognize as ones that can cause
1 !                                          confusion later is allowed to be
1 !                                          sent to terminal -- E.g. select graphics
1 !                                          rendition, etc ?
1 !
1 !                   10              Character can be discarded.  Character is
1 !                                   treated as a no-op.  It is broken out separately
1 !                                   in case we ever need to do something special
1 !                                   with it.
1 !
1 ! In summary:
1 !
1 !   Hex Character Codes    ASCII Character             Action Code
1 !   -------------------    ---------------             -----------
1 !       00 to 06           NUL to ACK                       1
1 !          07              BEL                              2
1 !          08              BS                               3
1 !          09              HT                               4
1 !          0A              LF                               5
1 !          0B              VT                               6
1 !          0C              FF                               7
1 !          0D              CR                               8
1 !       0E to 0F           SO to SI                         9
1 !       10 to 1A           DLE to SUB                       1
1 !          1B              ESC                              9
1 !       1C to 1F           FS to US                         1
1 !       20 to 7E           SP to ~                          0
1 !          7F              DEL                             10
1 !
1 !       80 to 9F           control chars                    1
1 !          A0              reserved                         1
1 !       A1 to FE           printing chars                   0
1 !          FF              reserved                         1
1 !
```

```
1 GLOBAL
1     CHAR_TABLE : VECTOR [256, BYTE] INITIAL ( BYTE (
1                                             ! 1st half is US ASCII
1                                             ! for DEC Multinational set (default)
1           1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, ! 00 to 0F
1           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, 1, 1, 1, 1, ! 10 to 1F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! 20 to 2F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! 30 to 3F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! 40 to 4F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! 50 to 5F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! 60 to 6F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,10, ! 70 to 7F
1                                             ! 2nd half is DEC Supplemental Graphics
1                                             ! for DEC Multinational set (default)
1           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ! 80 to 8F
1           1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ! 90 to 9F
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! A0 to AF
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! B0 to BF
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! C0 to CF
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! D0 to DF
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ! E0 to EF
1           0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0  ! F0 to FF
1                                                        ));
1
1
1
1
1 !<BLF/PAGE>
```

```
1 %SBTTL 'SMG$$PUT_TEXT_TO_BUFFER - Put text to buffer'
1 GLOBAL ROUTINE SMG$$PUT_TEXT_TO_BUFFER (
1                                 DCB : REF BLOCK [,BYTE],
1                                 ATTR_CODE : BYTE,
1                                 TEXT_LEN,
1                                 TEXT_ADDR,
1                                 CHAR_SET,
1                                 OVERFLOW
1                                         ) =
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This procedure places a text string into a buffer given the
1 !       current row and column in the buffer where output is to go.
1 !       The input text string is scanned for special characters that
1 !       prohibit simply moving the text into the buffer.  For example,
1 !       TABs reposition the maintained cursor position and the text
1 !       must be deposited at the appropriate tab boundaries as a
1 !       function of current position in the line.  Escape sequences
1 !       are not handled; an escape character is treated as a terminator,
1 !       and a qualified success status will be returned to indicate
1 !       that truncation occurred.
1 !
1 !       Positions in BUFFER that are modified have the corresponding
1 !       positions in ATTR_BUFFER and CHAR_BUFFER set.
1 !
1 !
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       ret_status.wlc.v = SMG$$PUT_TEXT_TO_BUFFER (
1 !                                 DCB.mab.r,
1 !                                 ATTR_CODE.rb.v,
1 !                                 TEXT_LEN.rl.v,
1 !                                 TEXT_ADDR.rl.v,
1 !                                 CHAR_SET.rl.v
1 !                                 [,OVERFLOW.wl.r])
1 !
1 ! FORMAL PARAMETERS:
1 !
1 !       DCB.mab.r         Address of virtual display control block.
1 !                         Various fields from within in this block are
1 !                         are interrogated and/or updated.
1 !
1 !       ATTR_CODE.rb.v    Video rendition attribute code.
1 !                         Bit 0   Bold
1 !                         Bit 1   Reverse video
1 !                         Bit 2   Blinking
1 !                         Bit 3   Underscored
1 !
1 !       TEXT_LEN.rl.v     Length of text string
1 !
1 !       TEXT_ADDR.rl.v    Address of text string
1 !
1 !       CHAR_SET.rl.v     Character set to use.
1 !                                 SMG$C_UNITED_KINGDOM
1 !                                 SMG$C_ASCII
```

```
1 !                                     SMG$C_SPEC_GRAPHICS
1 !                                     SMG$C_ALT_CHAR
1 !                                     SMG$C_ALT_GRAPHICS
1 !
1 !       OVERFLOW.wl.r   Optional.  Address of longword in which
1 !                       to return the number of characters that
1 !                       did not fit on the line.
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! COMPLETION STATUS:
1 !
1 !       SS$_NORMAL      Normal successful completion
1 !
1 ! SIDE EFFECTS:
1 !
1 !       NONE
1 !--
1
2     BEGIN
2
2     BUILTIN
2         SCANC,
2         NULLPARAMETER;
2
2     LOCAL
2         TEXT_BUF : REF VECTOR [,BYTE],  ! Addr of text buffer
2         ATTR_BUF : REF VECTOR [,BYTE],  ! Addr of attr buffer
2         CHAR_BUF : REF VECTOR [,BYTE],  ! Addr of char set buffer
2         STATUS,          ! status of subroutine calls
2         WORK_OVERFLOW : INITIAL (0),      ! no. of overflow chars
2         BYTES_REMAINING,! No. of bytes in input string yet to be
2                         ! processed.
2         IN_POINTER;     ! Current pointer into input string
2
2     LITERAL
2         K_OVERFLOW_ARG = 6;
2
2     TEXT_BUF = .DCB [DCB_A_TEXT_BUF];
2     ATTR_BUF = .DCB [DCB_A_ATTR_BUF];
2     CHAR_BUF = .DCB [DCB_A_CHAR_SET_BUF];
2
2     BYTES_REMAINING = .TEXT_LEN;
2     IN_POINTER = .TEXT_ADDR;
2
2     WHILE .BYTES_REMAINING NEQ 0
2     DO
3         BEGIN   ! Overall loop
3         LOCAL
3             CHARS_TO_MOVE,          ! No. of characters to move on this
3                                     ! iteration
```

```
3                   PLACE_TO_MOVE,          ! Place to move from on this iteration
3                   NEW_BYTES_REMAINING,    ! No. of bytes remaining as returned
3                                           ! by SCANC
3                   ADDR_DIFF;              ! Addr of char in input stream whose
3                                           ! index into scanc table yields
3                                           ! non-zero code.
3               !+
3               ! See if any of the remaining input characters require special
3               ! treatment.
3               !-
3               SCANC ( BYTES_REMAINING,            ! No. of bytes remaining
3                       .IN_POINTER,                ! Current pointer to source
3                       CHAR_TABLE,                 ! Address of SCANC table
3                       ALLONES;                    ! Mask for ANDing
3                       NEW_BYTES_REMAINING,        ! New remaining no. of bytes
3                                                   ! including the byte which
3                                                   ! caused the instruction to
3                                                   ! halt.  Is zero only if all
3                                                   ! bytes did not satify search.
3                       ADDR_DIFF);                 ! Addr of char in input stream
3                                                   ! whose index into scanc table
3                                                   ! yields non-zero code.
3
3               CHARS_TO_MOVE = .BYTES_REMAINING - .NEW_BYTES_REMAINING;
3               PLACE_TO_MOVE = .IN_POINTER;
3               IN_POINTER = .IN_POINTER + .CHARS_TO_MOVE;
3               BYTES_REMAINING = .NEW_BYTES_REMAINING;
3
3               !+
3               ! Copy the appropriate number of characters into the text buffer
3               ! and the appropriate number of copies of the attribute code
3               ! into the attribute buffer.
3               !-
3               IF .CHARS_TO_MOVE NEQ 0
3               THEN
4                   BEGIN
4                   LOCAL
4                       INDEX,  ! 0-based index into BUFFER and ATTR_BUFFER.
4                       REMAINING_COLS;
4
4                   INDEX = $SMG$LINEAR ( .DCB [DCB_W_CURSOR_ROW], .DCB [DCB_W_CURSOR_COL]);
4
4                   REMAINING_COLS = .DCB [DCB_W_NO_COLS] - .DCB [DCB_W_CURSOR_COL] + 1;
4                   IF .CHARS_TO_MOVE GTR .REMAINING_COLS
4                   THEN                                    ! chars will overflow line
5                       BEGIN
5                       WORK_OVERFLOW = .BYTES_REMAINING +
5                                       (.CHARS_TO_MOVE - .REMAINING_COLS);
5                       CHARS_TO_MOVE = .REMAINING_COLS;
4                       END;
4
4                   !+
4                   ! Move text into buffer.
4                   !-
4                   CH$MOVE (.CHARS_TO_MOVE,                ! No. of chars
4                            .PLACE_TO_MOVE,                ! From
4                            TEXT_BUF [ .INDEX ] );         ! To
```

```
4
4               !+
4               ! Rewrite attribute bytes.  Normally the attributes are
4               ! passed to us, but for the 'autobended' case where escape
4               ! sequences are used, we should look at the default attributes
4               ! which may have been altered by SMG$$SIM_TERM.
4               !-
5               BEGIN
5               LOCAL
5                   WORK_ATTR;
5               WORK_ATTR = .ATTR_CODE;
5               IF .DCB [DCB_V_ALLOW_ESC]
5               THEN
5                   WORK_ATTR = .DCB [DCB_B_DEF_VIDEO_ATTR];
5               CH$FILL (.WORK_ATTR,                    ! Char. to replicate
5                        .CHARS_TO_MOVE,                ! No. of times
5                        ATTR_BUF [ .INDEX ] );         ! Destination
4               END;
4
4               !+
4               ! Write the character set bytes, if necessary.
4               !-
4               IF .CHAR_BUF EQL 0 AND
4                  .CHAR_SET NEQ SMG$C_ASCII
4               THEN
4                   0;       ! first char set - alloc buffer
4
4               IF .CHAR_BUF NEQ 0
4               THEN
4                   CH$FILL (.CHAR_SET,
4                            .CHARS_TO_MOVE,
4                            CHAR_BUF [.INDEX]);
4
4               !+
4               ! Adjust resulting cursor position.  Check for overflow.
4               !-
4               DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] +
4                                        .CHARS_TO_MOVE;
4               IF .DCB [DCB_W_CURSOR_COL] GTR .DCB [DCB_W_NO_COLS]
4               THEN
4                   DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
4
4               IF .WORK_OVERFLOW NEQ 0
4               THEN
4                   EXITLOOP;
3               END;
3
3
3           IF .NEW_BYTES_REMAINING EQL 0
3           THEN
3               EXITLOOP;           ! Break out of loop -- we're done
3
3           !+
3           ! Dispatch on the non-zero code located to see what special
3           ! action is needed.
3           !-
3           CASE .CHAR_TABLE [.(.ADDR_DIFF) <0,8>] FROM 1 TO 10 OF
```

```
        SET
            [1]:
                !+
                !   Hex Character Codes       ASCII Character
                !   -------------------       ---------------
                !   00 to 06                   NUL to ACK
                !   10 to 1A                   DLE to SUB
                !   1C to 1F                   FS to US
                !
                ! Character can be discarded.  Cursor is not advanced.
                !
                ! Special case if the user_graphic bit is set.  That indicates
                ! a device-independent code which should be placed in the buffer
                ! for later interpretation by output.  Notice that we are guaranteed
                ! that TEXT_ADDR contains only 1 character since only we call this
                ! routine.
                !-
                IF (.ATTR_CODE AND ATTR_M_USER_GRAPHIC) NEQ 0
                THEN
                    $INSERT_CTRL_CHAR (.TEXT_ADDR);

            [2]:
                !+
                !   Hex Character Codes       ASCII Character
                !   -------------------       ---------------
                !      07                      BEL
                !
                ! Character can be discarded.  Cursor is not modified, and we
                ! call a routine to ring the bell now.  (Note that if we had
                ! stored the bell in the attribute buffer, the bell would've
                ! been rung every time the screen was repainted.)
                !-
                SMG$RING_BELL (.DCB [DCB_L_DID]);

            [3]:
                !+
                !   Hex Character Codes       ASCII Character
                !   -------------------       ---------------
                !      08                      BS
                !
                ! Character can be discarded, but cursor must be backed up
                ! one column.  Be careful about cursor already being in
                ! column 1.
                !-
                BEGIN
                IF .DCB [DCB_W_CURSOR_COL] NEQ 1
                THEN
                    DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_CURSOR_COL] -1;

                END;

            [4]:
                !+
                !   Hex Character Codes       ASCII Character
                !   -------------------       ---------------
                !      09                      HT
                !
```

```
3                  ! Character can be discarded, but cursor must be advanced to
3                  ! next TAB stop and intervening character positions in the
3                  ! buffer must be left undisturbed.
3                  !
3                  ! TAB stops are assumed to be set in the following columns:
3                  ! 9, 17, 25, 33, 41, 49, 57, 65, 73 ( width=80)
3                  !
3                  ! 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113,
3                  ! 121, 129 ( width=132)
3                  !-
4                  BEGIN
4                  !+
4                  ! Be careful about tabbing off the end of the line or beyond
4                  ! the end of the virtual display line.
4                  !-
4                  IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
4                  THEN
5                      BEGIN
5                      DCB [DCB_W_CURSOR_COL] =
5                                  ((.DCB [DCB_W_CURSOR_COL]-1)/8+1)*8+1;
5                      IF .DCB [DCB_W_CURSOR_COL] GTR .DCB [DCB_W_NO_COLS]
5                      THEN
5                          DCB [DCB_W_CURSOR_COL] = .DCB [DCB_W_NO_COLS];
5                      END
4                  ELSE
4                      $INSERT_CTRL_CHAR (TAB);
3                  END;
3
3                  [5,6]:
3                  !+
3                  !   Hex Character Codes        ASCII Character
3                  !   -------------------        ---------------
3                  !
3                  !      0A                          LF
3                  !      0B                          VT
3                  !
3                  ! Character can be discarded.  Cursor must be advanced by
3                  ! one line.  Don't advance beyond last line of display.
3                  !-
4                  BEGIN
4                  !+
4                  ! If cursor not at bottom, advance DCB [DCB_W_CURSOR_ROW]
4                  ! by one.
4                  !-
4                  IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
4                  THEN
5                      BEGIN
5                      IF .DCB [DCB_W_CURSOR_ROW] + 1 LEQ .DCB [DCB_W_BOTTOM_OF_SCRREG]
5                      THEN
5                          DCB [DCB_W_CURSOR_ROW] = .DCB [DCB_W_CURSOR_ROW] + 1
5                      ELSE
5                          SMG$$SCROLL_AREA (.DCB,
5                                      .DCB [DCB_W_TOP_OF_SCRREG],
5                                      .DCB [DCB_W_COL_START],
6                                      (.DCB [DCB_W_BOTTOM_OF_SCRREG] -
5                                       .DCB [DCB_W_TOP_OF_SCRREG] + 1),
5                                      .DCB [DCB_W_NO_COLS],
5                                      SMG$M_UP,
5                              1);
5         END
4     ELSE
5         BEGIN
5         LOCAL
5             CHAR;
5         CHAR = .(.ADDR_DIFF)<0,8>;
5         $INSERT_CTRL_CHAR (.CHAR);
4         END;
3     END;
3
3     [7]:
3     !+
3     !   Hex Character Codes        ASCII Character
3     !   -------------------        ---------------
3     !      0C                        FF
3     !
3     ! Character can be discarded.  Effect is to clear the buffer
3     ! and reset the cursor to line 1 column 1.
3     !-
4     BEGIN
4     IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
4     THEN
5         BEGIN
5         IF .DCB [DCB_W_CURSOR_ROW] + 1 LEQ .DCB [DCB_W_BOTTOM_OF_SCRREG]
5         THEN
5             DCB [DCB_W_CURSOR_ROW] = .DCB [DCB_W_CURSOR_ROW] + 1
5         ELSE
5             SMG$$SCROLL_AREA (.DCB,
5                            .DCB [DCB_W_TOP_OF_SCRREG],
5                            .DCB [DCB_W_COL_START],
6                            (.DCB [DCB_W_BOTTOM_OF_SCRREG] -
5                             .DCB [DCB_W_TOP_OF_SCRREG] + 1),
5                            .DCB [DCB_W_NO_COLS],
5                            SMG$M_UP,
5                            1);
5         END
4     ELSE
4         $INSERT_CTRL_CHAR (FF);
3     END;
3
3     [8]:
3     !+
3     !   Hex Character Codes        ASCII Character
3     !   -------------------        ---------------
3     !      0D                        CR
3     !
3     ! Character can be discarded.  Effect is to set cursor to
3     ! column 1 of current line.
3     !-
4     BEGIN
4     IF NOT .DCB [DCB_V_DISPLAY_CONTROLS]
4     THEN
4         DCB [DCB_W_CURSOR_COL] = 1
4     ELSE
4         $INSERT_CTRL_CHAR (CR);
3     END;
```

```
[9]:
!+
!   Hex Character Codes       ASCII Character
!   -------------------       ---------------
!      1B                     ESC
!      0E                     SO
!      0F                     SI
!
! Character can be discarded.  Subsequent characters need
! to be inspected to see if they constitute a recognized
! escape sequence whose effect must be simulated-- E.g.,
! cursor setting, rendition setting.
!
! SMG$$SIM_TERM processes the escape sequence, then returns
! here to allow any remaining characters to be processed.
!-
BEGIN
IF NOT .DCB [DCB_V_ALLOW_ESC]
THEN
    RETURN (SMG$_STRTERESC) ! error from true SMG$
ELSE
    BEGIN                      ! autobended - attempt to interpret
    LOCAL
        LEN_OF_SEQUENCE,
        STATUS;
    STATUS = SMG$$SIM_TERM (.DCB,
                           .BYTES_REMAINING,
                           .IN_POINTER,   ! pass ptr to esc char
                           LEN_OF_SEQUENCE);
    IF NOT .STATUS THEN RETURN (.STATUS);

    !+
    ! Update the number of bytes processed.  Since there is
    ! an automatic update (by 1 character) at the end of this
    ! loop, don't count the ESC now.
    !-
    BYTES_REMAINING = .BYTES_REMAINING - .LEN_OF_SEQUENCE + 1;
    IN_POINTER = .IN_POINTER + .LEN_OF_SEQUENCE - 1;
    END;                       ! autobended - attempt to interpret

END;

[10]:
!+
!   Hex Character Codes       ASCII Character
!   -------------------       ---------------
!      7F                      DEL
!
! Character can be discarded.
!-
;   ! no special action

[INRANGE, OUTRANGE]:
!+
! Should never get here -- there are no other codes in
! CHAR_TABLE.  If we do, we've got a problem.
```

```
;       3                   !-
;       4                   BEGIN
;       4                   RETURN SMG$_FATERRLIB;
;       3                   END;
;       3               TES;
;       3
;       3
;       3               !+
;       3               ! Re-adjust pointer and count of bytes left to account for
;       3               ! the special character(s) just processed.
;       3               !-
;       3               IN_POINTER = .IN_POINTER + 1;
;       3               BYTES_REMAINING = .BYTES_REMAINING -1;
;       2               END;    ! Overall loop
;       2
;       2           IF .DCB [DCB_W_CURSOR_COL] EQL .DCB [DCB_W_NO_COLS]
;       2           THEN
;       2               DCB [DCB_V_COL_80] = 1;
;       2
;       2           IF NOT NULLPARAMETER (K_OVERFLOW_ARG)
;       2           THEN
;       2               .OVERFLOW = .WORK_OVERFLOW;
;       2                                       ! ret overflow chars if requested
;       2           RETURN (SS$_NORMAL);
;       1           END;                        ! End of routine SMG$$PUT_TEXT_TO_BUFFER

                                                                .TITLE  SMG$$PUT_TEXT_TO_BUFFER Put text to display buf
;                                                                                               fer
                                                                .IDENT  \1-012\

                                                                .PSECT  _SMG$DATA,NOEXE,  PIC,2

                                                 FF  00000 ALLONES:.BYTE   -1                                                  ;
                                                     00001        .BLKB    3
09 08 07 06 05 04 03 02 01 01 01 01 01 01 01  00004 CHAR_TABLE::
                                                                .BYTE   1, 1, 1, 1, 1, 1, 1, 2, 3, 4, 5, 6, 7, 8, -  ;
01 01 09 01 01 01 01 01 01 01 01 01 01 01 09  00013                     9, 9, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 9, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 01 01  00022                     1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00031                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  00040                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0004F                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0005E                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  0006D                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
01 01 01 01 01 01 01 0A 00 00 00 00 00 00 00  0007C                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  0008B                     0, 10, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, - ;
00 00 00 00 00 01 01 01 01 01 01 01 01 01 01  0009A                     1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000A9                     1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000B8                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000C7                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000D6                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000E5                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  000F4                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
                                           00  00103                     0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -  ;
                                                                        0, 0, 0, 0, 0                                 ;

                                                                .EXTRN  SMG$$SIM_TERM, SMG$$SCROLL_AREA
```

```
                                                      N 14
SMG$$PUT_TEXT_T Put text to display buffer            16-Sep-1984 01:12:44    VAX-11 Bliss-32 V4.0-742             Page  17
1-012           SMG$$PUT_TEXT_TO_BUFFER - Put text to buffer    14-Sep-1984 13:10:00    [SMGRTL.SRC]SMGPUTTEX.B32;1           (5)

                                                            .EXTRN  SMG$RING_BELL, SMG$_FATERRLIB
                                                            .EXTRN  SMG$_STRTERESC

                                                            .PSECT  _SMG$CODE,NOWRT,  SHR,  PIC,2

                                            0FFC 00000      .ENTRY  SMG$$PUT_TEXT_TO_BUFFER, Save R2,R3,R4,R5,-  ; 0367
                                                                    R6,R7,R8,R9,R10,R11                          ;
                          5E              20   C2 00002     SUBL2   #32, SP                                      ;
                                    18    AE   D4 00005     CLRL    WORK_OVERFLOW                                ; 0448
                          59        04    AC   D0 00008     MOVL    DCB, R9                                      ; 0467
                          5B        10    A9   D0 0000C     MOVL    16(R9), TEXT_BUF                             ;
                          5A        14    A9   D0 00010     MOVL    20(R9), ATTR_BUF                             ; 0468
                     14   AE        18    A9   D0 00014     MOVL    24(R9), CHAR_BUF                             ; 0469
                          57        0C    AC   7D 00019     MOVQ    TEXT_LEN, BYTES_REMAINING                    ; 0471
                                          57   D5 0001D 1$: TSTL    BYTES_REMAINING                              ; 0474
                                          03   12 0001F     BNEQ    2$                                           ;
                                        02CE   31 00021     BRW     41$                                          ;
00000000'  EF 00000000'  EF     68        57   2A 00024 2$: SCANC   BYTES_REMAINING, (IN_POINTER), CHAR_TABLE, -; 0490
                                                                    ALLONES                                      ;
                     10   AE              50   D0 00031     MOVL    R0, 16(SP)                                   ;
                     0C   AE              51   D0 00035     MOVL    R1, 12(SP)                                   ;
               56         57        10    AE   C3 00039     SUBL3   NEW_BYTES_REMAINING, BYTES_REMAINING, -      ; 0503
                                                                    CHARS_TO_MOVE                                ;
                          52              58   D0 0003E     MOVL    IN_POINTER, PLACE_TO_MOVE                    ; 0504
                          58              56   C0 00041     ADDL2   CHARS_TO_MOVE, IN_POINTER                    ; 0505
                          57        10    AE   D0 00044     MOVL    NEW_BYTES_REMAINING, BYTES_REMAINING         ; 0506
                                          56   D5 00048     TSTL    CHARS_TO_MOVE                                ; 0513
                                          7C   13 0004A     BEQL    8$                                           ;
                          50        28    A9   3C 0004C     MOVZWL  40(R9), R0                                   ; 0520
                                          50   D7 00050     DECL    R0                                           ;
                          51        06    A9   3C 00052     MOVZWL  6(R9), R1                                    ;
                          50              51   C4 00056     MULL2   R1, R0                                       ;
                     08   AE        2A    A9   9E 00059     MOVAB   42(R9), 8(SP)                                ;
                          51        08    BE   3C 0005E     MOVZWL  @8(SP), R1                                   ;
                     04   AE        FF  A140   9E 00062     MOVAB   -1(R1)[R0], INDEX                            ;
                          50        06    A9   3C 00068     MOVZWL  6(R9), R0                                    ; 0522
                          50              51   C2 0006C     SUBL2   R1, R0                                       ;
                                          50   D6 0006F     INCL    REMAINING_COLS                               ;
                          50              56   D1 00071     CMPL    CHARS_TO_MOVE, REMAINING_COLS                ; 0523
                                          0C   15 00074     BLEQ    3$                                           ;
               51         56              50   C3 00076     SUBL3   REMAINING_COLS, CHARS_TO_MOVE, R1            ; 0527
          18   AE         51              57   C1 0007A     ADDL3   BYTES_REMAINING, R1, WORK_OVERFLOW           ;
                          56              50   D0 0007F     MOVL    REMAINING_COLS, CHARS_TO_MOVE                ; 0528
          04 BE4B         62              56   28 00082 3$: MOVC3   CHARS_TO_MOVE, (PLACE_TO_MOVE), @INDEX-      ; 0536
                                                                    [TEXT_BUF]                                   ;
                          50        08    AC   9A 00088     MOVZBL  ATTR_CODE, WORK_ATTR                         ; 0547
               04    34   A9              05   E1 0008C     BBC     #5, 52(R9), 4$                               ; 0548
                          50        2E    A9   9A 00091     MOVZBL  46(R9), WORK_ATTR                            ; 0550
     56        50         6E              00   2C 00095 4$: MOVC5   #0, (SP), WORK_ATTR, CHARS_TO_MOVE, @INDEX-  ; 0553
                                    04  BE4A      0009A             [ATTR_BUF]                                   ;
                                    14    AE   D5 0009D     TSTL    CHAR_BUF                                     ; 0559
                                          0E   13 000A0     BEQL    5$                                           ; 0564
               6E    04   AE        14    AE   C1 000A2     ADDL3   CHAR_BUF, INDEX, (SP)                        ; 0568
     56   14   AC         6E              00   2C 000A8     MOVC5   #0, (SP), CHAR_SET, CHARS_TO_MOVE, @0(SP)    ;
                                    00    BE      000AE                                                          ;
                     08   BE              56   A0 000B0 5$: ADDW2   CHARS_TO_MOVE, @8(SP)                        ; 0574
                     06   A9        08    BE   B1 000B4     CMPW    @8(SP), 6(R9)                                ; 0575
```

```
                                     05   1B 000B9        BLEQU   6$                                  ;
                    08   BE      06  A9   B0 000BB        MOVW    6(R9), @8(SP)                       ; 0577
                                 18  AE   D5 000C0 6$:    TSTL    WORK_OVERFLOW                       ; 0579
                                     03   13 000C3        BEQL    8$                                  ;
                                   022A   31 000C5 7$:    BRW     41$                                 ;
                                 10  AE   D5 000C8 8$:    TSTL    NEW_BYTES_REMAINING                 ; 0585
                                     F8   13 000CB        BEQL    7$                                  ;
                         52      0C  BE   9A 000CD        MOVZBL  @ADDR_DIFF, R2                      ; 0593
              09         01 00000000'EF42 8F 000D1        CASEB   CHAR_TABLE[R2], #1, #9              ;
    006C    0061       0055        001C      000DA 9$:    .WORD   10$-9$,-                            ;
    0186    0118       00DA        00DA      000E2                12$-9$,-                            ;
                       0211        01E0      000EA                13$-9$,-                            ;
                                                                  15$-9$,-                            ;
                                                                  21$-9$,-                            ;
                                                                  21$-9$,-                            ;
                                                                  23$-9$,-                            ;
                                                                  31$-9$,-                            ;
                                                                  38$-9$,-                            ;
                                                                  40$-9$                              ;
                         50 00000000G  8F D0 000EE        MOVL    #SMG$_FATERRLIB, R0                 ; 0824
                                         04 000F5         RET                                         ;
              49    08   AC          06  E1 000F6 10$:    BBC     #6, ATTR_CODE, 14$                  ; 0611
                         53      06  A9  3C 000FB         MOVZWL  6(R9), REMAINING_COLS               ; 0613
                         50      28  A9  3C 000FF         MOVZWL  40(R9), R0                          ;
                         53          50  C2 00103         SUBL2   R0, REMAINING_COLS                  ;
                         50      28  A9  3C 00106         MOVZWL  40(R9), R0                          ;
                                     50  D7 0010A         DECL    R0                                  ;
                         51      06  A9  3C 0010C         MOVZWL  6(R9), R1                           ;
                         50          51  C4 00110         MULL2   R1, R0                              ;
                         52      2A  A9  9E 00113         MOVAB   42(R9), R2                          ;
                         51          62  3C 00117         MOVZWL  (R2), R1                            ;
                         51   FF A140  9E 0011A           MOVAB   -1(R1)[R0], INDEX                   ;
                                     53  D5 0011F         TSTL    REMAINING_COLS                      ;
                                     03  14 00121         BGTR    11$                                 ;
                                   016C  31 00123         BRW     34$                                 ;
    53      10   AC        04        00  EF 00126 11$:    EXTZV   #0, #4, TEXT_ADDR, R3               ;
                                   00BA  31 0012C         BRW     22$                                 ;
                                 38  A9  DD 0012F 12$:    PUSHL   56(R9)                              ; 0626
                 00000000G  00       01  FB 00132         CALLS   #1, SMG$RING_BELL                   ;
                                     77  11 00139         BRB     20$                                 ;
                         01      2A  A9  B1 0013B 13$:    CMPW    42(R9), #1                          ; 0639
                                     71  13 0013F         BEQL    20$                                 ;
                                 2A  A9  B7 00141         DECW    42(R9)                              ; 0641
                                     6C  11 00144 14$:    BRB     20$                                 ; 0593
                         53      2A  A9  9E 00146 15$:    MOVAB   42(R9), R3                          ; 0669
              18    2F   A9          02  E0 0014A         BBS     #2, 47(R9), 16$                     ; 0666
                         50          63  3C 0014F         MOVZWL  (R3), R0                            ; 0670
                                     50  D7 00152         DECL    R0                                  ;
                         50          08  C6 00154         DIVL2   #8, R0                              ;
              51         50          03  78 00157         ASHL    #3, R0, R1                          ;
              63         51          09  A1 0015B         ADDW3   #9, R1, (R3)                        ;
                    06   A9          63  B1 0015F         CMPW    (R3), 6(R9)                         ; 0671
                                     4D  1B 00163         BLEQU   20$                                 ;
                                     47  11 00165         BRB     19$                                 ; 0673
                         52      06  A9  3C 00167 16$:    MOVZWL  6(R9), REMAINING_COLS               ; 0676
                         50      28  A9  3C 0016B         MOVZWL  40(R9), R0                          ;
                         52          50  C2 0016F         SUBL2   R0, REMAINING_COLS                  ;
```

```
                                                            C 15
SMG$$PUT_TEXT_T Put text to display buffer                  16-Sep-1984 01:12:44    VAX-11 Bliss-32 V4.0-742                  Page  19
1-012           SMG$$PUT_TEXT_TO_BUFFER - Put text to buffer    14-Sep-1984 13:10:00    [SMGRTL.SRC]SMGPUTTEX.B32;1                (5)

                             50     28   A9  3C 00172        MOVZWL  40(R9), R0
                                         50  D7 00176        DECL    R0
                             51     06   A9  3C 00178        MOVZWL  6(R9), R1
                             50          51  C4 0017C        MULL2   R1, R0
                             51          63  3C 0017F        MOVZWL  (R3), R1
                             51  FF A140     9E 00182        MOVAB   -1(R1)[R0], INDEX
                                         52  D5 00187        TSTL    REMAINING_COLS
                                         06  14 00189        BGTR    17$
                      18     AE          57  D0 0018B        MOVL    BYTES_REMAINING, WORK_OVERFLOW
                                         15  11 0018F        BRB     18$
                             50     90   8F  90 00191 17$:   MOVB    #-112, SHIFT_NIBBLE
                           614B          50  90 00195        MOVB    SHIFT_NIBBLE, (INDEX)[TEXT_BUF]
                             50     08   AC  9A 00199        MOVZBL  ATTR_CODE, WORK_ATTR
50          01               06          01  F0 0019D        INSV    #1, #6, #1, WORK_ATTR
                           614A          50  90 001A2        MOVB    WORK_ATTR, (INDEX)[ATTR_BUF]
                                         63  B6 001A6 18$:   INCW    (R3)
                      06     A9          63  B1 001A8        CMPW    (R3), 6(R9)
                                         5A  12 001AC        BNEQ    25$
                             63     06   A9  B0 001AE 19$:   MOVW    6(R9), (R3)
                                         7D  11 001B2 20$:   BRB     27$                                    ; 0593
            3E        2F     A9          02  E1 001B4 21$:   BBC     #2, 47(R9), 24$                        ; 0694
                             54          52  D0 001B9        MOVL    R2, CHAR                               ; 0714
                             53     06   A9  3C 001BC        MOVZWL  6(R9), REMAINING_COLS                  ; 0715
                             50     28   A9  3C 001C0        MOVZWL  40(R9), R0
                             53          50  C2 001C4        SUBL2   R0, REMAINING_COLS
                             50     28   A9  3C 001C7        MOVZWL  40(R9), R0
                                         50  D7 001CB        DECL    R0
                             51     06   A9  3C 001CD        MOVZWL  6(R9), R1
                             50          51  C4 001D1        MULL2   R1, R0
                             52     2A   A9  9E 001D4        MOVAB   42(R9), R2
                             51          62  3C 001D8        MOVZWL  (R2), R1
                             51  FF A140     9E 001DB        MOVAB   -1(R1)[R0], INDEX
                                         53  D5 001E0        TSTL    REMAINING_COLS
                                         75  15 001E2        BLEQ    29$
53          54               04          00  EF 001E4        EXTZV   #0, #4, CHAR, R3
            53               53          04  78 001E9 22$:   ASHL    #4, R3, R3
                             50          53  90 001ED        MOVB    R3, SHIFT_NIBBLE
                                         6C  11 001F0        BRB     30$
            3C        2F     A9          02  E0 001F2 23$:   BBS     #2, 47(R9), 28$                        ; 0729
                             50     28   A9  3C 001F7 24$:   MOVZWL  40(R9), R0                             ; 0732
                                         50  D6 001FB        INCL    R0
50      4A  A9               10          00  ED 001FD        CMPZV   #0, #16, 74(R9), R0
                                         05  19 00203        BLSS    26$
                                    28   A9  B6 00205        INCW    40(R9)                                 ; 0734
                                         62  11 00208 25$:   BRB     32$
                                         01  DD 0020A 26$:   PUSHL   #1                                     ; 0736
                                         01  DD 0020C        PUSHL   #1
                             7E     06   A9  3C 0020E        MOVZWL  6(R9), -(SP)                           ; 0741
                             50     4A   A9  3C 00212        MOVZWL  74(R9), R0                             ; 0740
                             51     48   A9  3C 00216        MOVZWL  72(R9), R1
                             50          51  C2 0021A        SUBL2   R1, R0
                                    01   A0  9F 0021D        PUSHAB  1(R0)                                  ; 0739
                             7E     04   A9  3C 00220        MOVZWL  4(R9), -(SP)                           ; 0738
                             7E     48   A9  3C 00224        MOVZWL  72(R9), -(SP)                          ; 0737
                                         59  DD 00228        PUSHL   R9                                     ; 0736
               00000000G     00          07  FB 0022A        CALLS   #7, SMG$$SCROLL_AREA
                                         39  11 00231 27$:   BRB     32$                                    ; 0729
                                    53          06    A9  3C 00233 28$:   MOVZWL  6(R9), REMAINING_COLS                     ; 0746
                                    50          28    A9  3C 00237        MOVZWL  40(R9), R0                                ;
                                    53                50  C2 0023B        SUBL2   R0, REMAINING_COLS                        ;
                                    50          28    A9  3C 0023E        MOVZWL  40(R9), R0                                ;
                                                      50  D7 00242        DECL    R0                                        ;
                                    51          06    A9  3C 00244        MOVZWL  6(R9), R1                                 ;
                                    50                51  C4 00248        MULL2   R1, R0                                    ;
                                    52          2A    A9  9E 0024B        MOVAB   42(R9), R2                                ;
                                    51                62  3C 0024F        MOVZWL  (R2), R1                                  ;
                                    51          FF  A140  9E 00252        MOVAB   -1(R1)[R0], INDEX                         ;
                                                      53  D5 00257        TSTL    REMAINING_COLS                            ;
                                                      37  15 00259 29$:   BLEQ    34$                                       ;
                                    50                3F  92 0025B        MCOMB   #63, SHIFT_NIBBLE                         ;
                                                      3B  11 0025E 30$:   BRB     36$                                       ;
                                    52          2A    A9  9E 00260 31$:   MOVAB   42(R9), R2                                ; 0761
                      05        2F  A9                02  E0 00264        BBS     #2, 47(R9), 33$                           ; 0759
                                    62                01  B0 00269        MOVW    #1, (R2)                                  ; 0761
                                                      7D  11 0026C 32$:   BRB     40$                                       ;
                                    53          06    A9  3C 0026E 33$:   MOVZWL  6(R9), REMAINING_COLS                     ; 0763
                                    50          28    A9  3C 00272        MOVZWL  40(R9), R0                                ;
                                    53                50  C2 00276        SUBL2   R0, REMAINING_COLS                        ;
                                    50          28    A9  3C 00279        MOVZWL  40(R9), R0                                ;
                                                      50  D7 0027D        DECL    R0                                        ;
                                    51          06    A9  3C 0027F        MOVZWL  6(R9), R1                                 ;
                                    50                51  C4 00283        MULL2   R1, R0                                    ;
                                    51                62  3C 00286        MOVZWL  (R2), R1                                  ;
                                    51          FF  A140  9E 00289        MOVAB   -1(R1)[R0], INDEX                         ;
                                                      53  D5 0028E        TSTL    REMAINING_COLS                            ;
                                                      06  14 00290        BGTR    35$                                       ;
                                18  AE                57  D0 00292 34$:   MOVL    BYTES_REMAINING, WORK_OVERFLOW            ;
                                                      14  11 00296        BRB     37$                                       ;
                                    50                30  8E 00298 35$:   MNEGB   #48, SHIFT_NIBBLE                         ;
                                  614B                50  90 0029B 36$:   MOVB    SHIFT_NIBBLE, (INDEX)[TEXT_BUF]           ;
                                    50          08    AC  9A 0029F        MOVZBL  ATTR_CODE, WORK_ATTR                      ;
          50          01            06                01  F0 002A3        INSV    #1, #6, #1, WORK_ATTR                     ;
                                  614A                50  90 002A8        MOVB    WORK_ATTR, (INDEX)[ATTR_BUF]              ;
                                                      62  B6 002AC 37$:   INCW    (R2)                                      ;
                                06  A9                62  B1 002AE        CMPW    (R2), 6(R9)                               ;
                                                      37  12 002B2        BNEQ    40$                                       ;
                                    62          06    A9  B0 002B4        MOVW    6(R9), (R2)                               ;
                                                      31  11 002B8        BRB     40$                                       ; 0593
                      08        34  A9                05  E0 002BA 38$:   BBS     #5, 52(R9), 39$                           ; 0783
                                    50   00000000G    8F  D0 002BF        MOVL    #SMG$_STRTERESC, R0                       ; 0785
                                                      04  002C6           RET                                               ;
                                                1C    AE  9F 002C7 39$:   PUSHAB  LEN_OF_SEQUENCE                           ; 0791
                                    7E                57  7D 002CA        MOVQ    BYTES_REMAINING, -(SP)                    ; 0792
                                                      59  DD 002CD        PUSHL   R9                                        ; 0791
                         00000000G  00                04  FB 002CF        CALLS   #4, SMG$$SIM_TERM                         ;
                                    36                50  E9 002D6        BLBC    STATUS, 44$                               ; 0795
                      50            57          1C    AE  C3 002D9        SUBL3   LEN_OF_SEQUENCE, BYTES_REMAINING, R0      ; 0802
                                    57          01    A0  9E 002DE        MOVAB   1(R0), BYTES_REMAINING                    ;
                      50            58          1C    AE  C1 002E2        ADDL3   LEN_OF_SEQUENCE, IN_POINTER, R0           ; 0803
                                    58          FF    A0  9E 002E7        MOVAB   -1(R0), IN_POINTER                        ;
                                                      58  D6 002EB 40$:   INCL    IN_POINTER                                ; 0833
                                                      57  D7 002ED        DECL    BYTES_REMAINING                           ; 0834
                                                FD2B      31 002EF        BRW     1$                                        ; 0474
                                06  A9          2A    A9  B1 002F2 41$:   CMPW    42(R9), 6(R9)                             ; 0837
```

```
                                    04  12 002F7          BNEQ    42$                                          ;
                  34   A9           02  88 002F9          BISB2   #2, 52(R9)                                   ; 0839
                       06           6C  91 002FD 42$:     CMPB    (AP), #6                                     ; 0841
                                    0A  1F 00300          BLSSU   43$                                          ;
                               18   AC  D5 00302          TSTL    24(AP)                                       ;
                                    05  13 00305          BEQL    43$                                          ;
                  18   BC      18   AE  D0 00307          MOVL    WORK_OVERFLOW, @OVERFLOW                     ; 0843
                       50           01  D0 0030C 43$:     MOVL    #1, R0                                       ; 0845
                                        04 0030F 44$:     RET                                                  ; 0846

; Routine Size:  784 bytes,    Routine Base:  _SMG$CODE + 0000


;  774          0847  1 !<BLF/PAGE>
;  776          0848  1 END                          ! End of module SMG$$PUT_TEXT_TO_BUFFER
;  777          0849  1
;  778          0850  0 ELUDOM



;                                PSECT SUMMARY
;
;       Name                 Bytes                        Attributes
;
;   _SMG$DATA                     260  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,  PIC,ALIGN(2)
;   _SMG$CODE                     784  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)



;                          Library Statistics
;
;                                             -------- Symbols --------       Pages      Processing
;       File                                  Total    Loaded   Percent       Mapped     Time
;
;   _$255$DUA28:[SYSLIB]STARLET.L32;1          9776         5         0          581      00:01.0
;   _$255$DUA28:[SMGRTL.OBJ]RTLLIB.L32;1         36         0         0            8      00:00.1
;   _$255$DUA28:[SMGRTL.OBJ]SMGLIB.L32;1        469        19         4           38      00:00.4



;                                COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:SMGPUTTEX/OBJ=OBJ$:SMGPUTTEX MSRC$:SMGPUTTEX/UPDATE=(ENH$:SMGPUTTEX
;        )

; Size:          784 code + 260 data bytes
; Run Time:         00:23.5
; Elapsed Time:     01:25.0
; Lines/CPU Min:     2170
; Lexemes/CPU-Min:  18446
; Memory Used:   354 pages
; Compilation Complete
```

0360 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SMGNUMTAB
LIS

SMGMSGPTR
LIS

SMGSCROLL
LIS

SMGMISC
LIS

SMGMSGTXT
LIS

SMGPUTENC
LIS

SMGPUTTEX
LIS

SMGSIMTRM
LIS

SMGNUMPAR
LIS

SMGPRVINP
LIS